

LASER INTERFEROMETER GRAVITATIONAL WAVE OBSERVATORY
- LIGO -
CALIFORNIA INSTITUTE OF TECHNOLOGY
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

Document Type	LIGO-T030293-01-Z	April 29, 2004
MATLAB "Channel" Class Software Specification		
Keith Thorne		

Distribution of this draft:

California Institute of Technology
LIGO Project - MS 51-33
Pasadena, CA 91125
Phone (626) 395-2129
Fax (626) 304-9834
E-mail: info@ligo.caltech.edu

Massachusetts Institute of Technology
LIGO Project - MS 20B-145
Cambridge, MA 01239
Phone (617) 253-4824
Fax (617) 253-7014
E-mail: info@ligo.mit.edu

WWW: <http://www.ligo.caltech.edu/>

Contents

1	Introduction	3
2	Approach	3
3	Channel Class Specification	3
3.1	Requirements	3
3.2	Channel Class Interface	4
4	Implementation Plan	5
5	Channel Structure for MATLAB Compiler use	5
5.1	Channel Structure Interface	6
6	Additional capabilities for the Channel structure	6
6.1	Additional Channel Structure Interfaces	7

1 Introduction

A MATLAB class, termed "Channel", provides a way to accessing the time-series data from gravity-wave interferometers (such as LIGO) from within MATLAB which "abstracts" them as objects. In this way, the particular details of how the time-series data are obtained from LDAS, LDR or otherwise are hidden.

2 Approach

To implement this new access, a MATLAB class called "Channel" will be used. MATLAB classes are similar to C++ classes. This "Channel" class will be used to create objects with the properties of that class, which act a lot like structures. A software requirements specification was first created for the class. This specification forms the basis for the component tests to be used to verify an implementation of the software. Each 'shall' is intended to be a testable condition. This specification is followed by an explanation of the class interface to be created.

At present, the MATLAB Compiler does not support MATLAB object-oriented coding. To support this usage, a Channel structure similar in behavior to the class will also be implemented.

Once we had created the initial implementation and started using the software, it was desired that additional capabilities be added to the software, especially to the structure implementation. These capability requests were turned into additional requirements in the specification.

3 Channel Class Specification

The Channel class will provide an association to a particular interferometer data stream that hides exactly how that connection is provided. This 'abstraction' will allow the frame data retrieval method to be changed without requiring a change to the main analysis code.

3.1 Requirements

The software shall allow creation of Channel objects which are each associated with a specific Interferometric Gravity-Wave Detector (IGWD) channel name. The software will support the following properties for a Channel object: 'name', 'statusCode', 'site', 'instrument', 'type', 'gpsStart' and 'rate'. The 'name' property will be a string that the software shall set to the channel name used when creating the Channel object. The 'statusCode' property will be an integer that the software shall set to 0 if the Channel object was created successfully. The software shall not allow 'name' and 'statusCode' to be set externally. The 'site' property will be a string that corresponds to the LDAS 'sys' parameter. The 'instrument' property will be a string the corresponds to the instrument defined in document LIGO-T970130-E. The 'type' property will be a string that corresponds to the frame type defined in document LIGO-T970130-E. The software shall set the 'site' and 'type' properties to default values of 'LDR' and 'RDS_R_L1' when a Channel object is created. The software shall set the 'instrument' property to a default value appropriate to the 'name' of the Channel object when that object is created. The software shall allow the 'site', 'type' and 'instrument' properties to be set externally.

The 'gpsStart' property will be an integer in units of GPS seconds which is the starting location for all time series derived from this channel. The software shall set the 'gpsStart' property when a Channel object is created if a gpsStart is passed as a parameter. If no gpsStart parameter is passed when creating a Channel object, the software shall set the 'gpsStart' to a default value of 0. The software shall allow the 'gpsStart' property to be set externally. It is assumed that the IGWD data is stored in the format of frame files.

The 'rate' property will be an integer in units of samples per seconds. This represents the number of samples per second seen in data taken from frame files for this channel given the 'type' and 'instrument'. When the Channel object is created, a default value of 0 shall be set by the software.

The software will handle channel names of the format defined in Appendix D of document LIGO-T970130-E. If the channel name is not of this format when a Channel object is created, the software shall indicate the error by setting the 'statusCode' property to 1.

The software shall allow the creation of time-series data vectors from Channel objects. The software will allow time-series data to be created based upon inputs of 'gpsOffset' and either a 'duration' value or a 'gpsEnd' value. The 'gpsOffset' and 'gpsEnd' values will be in units of GPS seconds (either positive integer or positive real). The 'duration' value will be in units of seconds (either positive integer or positive real). The software shall start the time-series data vector at a GPS time of 'gpsOffset' + the 'gpsStart' property of the specific Channel object. The software shall create a time-series data vector which covers a range of 'duration' seconds. If the 'gpsEnd' input is provided instead of 'duration', the software shall set the duration to be 'gpsEnd' - 'gpsOffset'. The length of the time-series data vector will be 'duration' /times sample rate of the data available for the channel. When creating the time-series data vector, the software shall use the 'name', 'instrument' and 'type' properties of the specific Channel object.

A few error conditions are defined for attempt creation of time-series data vectors. If any of the input values are invalid (non-existent Channel object, gpsOffset missing, duration = 0, etc.), the software shall set the output vector to an empty array. If the software is not able to find frame data files which cover the time-frame requested (based upon gpsStart, gpsOffset and duration) for the values of 'name', 'instrument' and 'type' properties, then the software shall set the output vector to an empty array. If the software is not able to open a frame data file it has the path to, it shall set the output vector to an empty array.

Whenever the software creates a time-series data vector for a Channel object, it shall set the 'rate' property of the Channel object based upon the data read from the frame files processed to create the time-series data vector.

3.2 Channel Class Interface

The MATLAB syntax for creating a new Channel object will be

```
chanObj = Channel(nameString, <gpsStartTimeValue>)
```

Note that gpsStartTimeValue is an optional input. The properties can be retrieved using the 'get' function.

```
status = get(chanObj, 'statusCode')
```

Some properties can be changed using the 'set' function, which returns the new value of that property.

```
chanObj = set(chanObj, 'type', 'RAW')
```

The properties can also be retrieved using '.' notation.

```
status = chanObj.statusCode
```

Some properties can also be changed using the '.' notation.

```
chanObj.type='RAW'
```

The MATLAB syntax for getting a time-series data vector from a Channel object will be in one of two forms. Either the (s,d) format:

```
newSeries = chanObj(gpsOffset,duration)
```

where 'chanObj' is a previously-created Channel object, 'gpsOffset' is the offset of the start of the time-series from the 'gpsStart' property of the channel, and 'duration' is the length of the time-series in whole seconds. Or in the (s:e) format:

```
newSeries = chanObj(gpsOffset:gpsEnd)
```

where 'chanObj' is a previously-created Channel object, 'gpsOffset' is the offset of the start of the time-series from the 'gpsStart' property of the channel, and 'gpsEnd' defining the duration as 'gpsEnd' - 'gpsOffset'.

4 Implementation Plan

The initial implementation will support the only the LDR retrieval method using a database query (using MySQL) of the LDR file list. As such, the 'site' property will not be implemented at this time. If support for the LDAS Frame Cache query retrieval method is needed in the future, it will be implemented at that time.

5 Channel Structure for MATLAB Compiler use

As of MATLAB Compiler 3.0, MATLAB object-oriented code (i.e. classes) can not be present in compiled code used for stand-alone libraries and executables. To allow syntaxes similar to that used for the Channel class, the software will implement a Channel structure using the 'chanstruct' and 'chanvector' functions.

The 'chanstruct' function shall return a MATLAB structure with the same properties ('name', 'statusCode', 'site', 'instrument', 'type', 'gpsStart' and 'rate') as the Channel class. The 'chanvector' function shall return a time-series data vector which is identical to that retrieved from a Channel class object.

5.1 Channel Structure Interface

The MATLAB syntax for creating a Channel structure will be

```
chanStr = chanstruct(nameString, <gpsStartTimeValue>)
```

Note that `gpsStartTimeValue` is an optional input. The Channel structure properties can also be retrieved using `.'` notation.

```
status = chanStr.statusCode
```

The Channel structure properties can also be changed using the `.'` notation.

```
chanStr.type='RAW'
```

The MATLAB syntax for getting a time-series data vector from Channel structure will be in one of two forms. Either the (s,d) format:

```
newSeries = chanvector(chanStr, gpsOffset, duration)
```

where `'chanStr'` is a previously-created Channel structure, `'gpsOffset'` is the offset of the start of the time-series from the `'gpsStart'` property of the channel, and `'duration'` is the length of the time-series in whole seconds. Or in the (s:e) format:

```
newSeries = chanvector(chanStr, gpsOffset:gpsEnd)
```

where `'chanStr'` is a previously-created Channel object, `'gpsOffset'` is the offset of the start of the time-series from the `'gpsStart'` property of the channel, and `'gpsEnd'` defining the duration as `'gpsEnd' - 'gpsOffset'`. With the `'chanvector'` function, the sample rate (in samples/second) can also be retrieved as an optional output parameter.

```
[newSeries, sampRate] = chanvector(chanStr, gpsOffset, duration)
```

6 Additional capabilities for the Channel structure

It is desired that users be able to retrieve the list of physical frame files which cover a given time range for a Channel structure. It is also desired that users be able to supply an external list of physical frame files which creating a time-series data vector for a Channel structure. This would allow caching of frame files lists to combat delays from slow database queries and to easily use frame file sources not presently managed by LDR.

To these ends, the software shall allow a list of physical frame files (together with frame GPS start times and durations) to be retrieved for an time range of an existing Channel structure. The software will allow the optional listing of the physical frame files to be used when getting a time-series data vector for a Channel structure. If the optional frame file listing is provided, the software shall use that list of frame files and not use a list derived from its standard method.

It was also desired to allow the optional retrieval of time-series data from frames for time ranges which are not totally covered by existing frame files. If the `'rate'` variable is non-zero for a Channel structure when retrieving a time-series data vector, the software shall always return a data vector of the correct length. However, the software shall set all samples for which frame data was

not present equal to zero. If sufficient frame file data is present for the input time range to allow a determination of the sample rate, the software shall return a data vector of the correct length even if the 'rate' property is at the default value of 0 for the Channel structure. If samples of the time-series data vector have been set equal to zero to account for missing frame file data, the software shall set an optional status parameter to a non-zero value. The software shall set the optional status parameter to a zero value if there are no gaps in coverage from frame file data and the data from any particular frame file is not zero for all samples for that channel.

6.1 Additional Channel Structure Interfaces

The MATLAB syntax for retrieving the list of frame files covering a given time range for a Channel structure is similar to that to get the time-series data Either the (s,d) format:

```
[gpsTimes, frameFiles, frameDurs = chanframe(chanStr, gpsOffset, duration)
```

or (s:e) format:

```
[gpsTimes, frameFiles, frameDurs = chanframe(chanStr, gpsOffset:gpsEnd)
```

can be used. Here 'frameFiles' is a cell array of strings representing the full path to physical frame files covering the desired time range for the Channel. The 'gpsTimes' is an array of GPS start times (in integer seconds), one entry for each frame file in 'frameFiles'; The 'frameDurs' is an array of durations (in integer seconds), one entry for each frame files in 'frameFiles'; The MATLAB syntax for the external list of frame files when getting a time-series data vector is as follows:

```
newSeries = chanvector(chanStr, gpsOffset, duration, ...
gpsTimes, frameFiles, frameDurs)
```

```
newSeries = chanvector(chanStr, gpsOffset:gpsEnd, ...
gpsTimes, frameFiles, frameDurs)
```

The Matlab syntax for the optional status parameter is

```
[newSeries, sampRate, statusVar] = chanvector(chanStr, gpsOffset, duration)
```